



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/940,242	08/27/2001	Michael L. Van De Vanter	004-4911-1	7182

22120 7590 11/10/2005

ZAGORIN O'BRIEN GRAHAM LLP  
7600B N. CAPITAL OF TEXAS HWY.  
SUITE 350  
AUSTIN, TX 78731

EXAMINER

INGBERG, TODD D

ART UNIT PAPER NUMBER

2193

DATE MAILED: 11/10/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

<b>Office Action Summary</b>	<b>Application No.</b> 09/940,242	<b>Applicant(s)</b> VAN DE VANTER ET AL.	
	<b>Examiner</b> Todd Ingberg	<b>Art Unit</b> 2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

#### Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

#### Status

- 1) ☒ Responsive to communication(s) filed on 28 September 2005.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

#### Disposition of Claims

- 4) ☒ Claim(s) 1-30 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☐ Claim(s) \_\_\_\_\_ is/are allowed.
- 6) ☒ Claim(s) 1-30 is/are rejected.
- 7) ☐ Claim(s) \_\_\_\_\_ is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

#### Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 3/3/2003 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

#### Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
  - ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
  - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

\* See the attached detailed Office action for a list of the certified copies not received.

#### Attachment(s)

- |  |   |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892)  | 4) <input type="checkbox"/> Interview Summary (PTO-413)<br>Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)                                   | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152)             |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)<br>Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____  |

## **DETAILED ACTION**

In response to the After Final response filed September 28, 2005 prosecution is reopened.

Claims 1 – 30 have been examined.

Claim 1 has been amended.

### ***Interpretation Section***

1. This section is the understanding of the invention and the claim interpretation. The section is the result of an Interview.

#### **I. Actual Invention**

Functionality provided by the Invention is the ability for programmers to enter code into an IDE and the underlying interpreter distinguishes between code (Such as JAVA™) and the comments which are able to be formatted in to comments in a hypertext format. The example of JAVA and a Hypertext language is not a limiting example. The invention is the support to distinguish based on lexical rules. Which in short means the languages do not need to be explicitly claimed. The Examiner used these two as an example.

#### **II. Claimed Invention**

Claim 1 was discussed and is presented below:

Claim 1

An interactive software engineering tool, which is embodied on a computer readable medium, that, for distinct portions of a single unit of source code thereof with behavior according to a corresponding set of lexical rules, wherein transition of the behavior from that in accordance with a first lexical context to that in accordance with a second lexical context is based on recognition of an opening boundary token according to the first lexical context and without use of a structural command to the interactive software engineering tool.

Art Unit: 2193

**Claim Analysis**

**Claim Limitations**

An interactive software engineering tool, which is embodied on a computer readable medium, that,

**Meaning of Claim Limitations**

The preamble is given patentable weight.

**Claim Limitations**

for distinct portions of a single unit of source code

**Meaning of Claim Limitations**

The ability to enter code in the interactive software engineering tool.

**Claim Limitations**

thereof with behavior according to a corresponding set of lexical rules,

**Meaning of Claim Limitations**

The code has lexical rules which inherently relate to grammar rules.

**Claim Limitations**

wherein transition of the behavior from that in accordance with a first lexical context to that in accordance with a second lexical context is based on recognition of an opening boundary token

**Meaning of Claim Limitations**

The ability to transition between two distinct lexical contexts (example given JAVA and comments).

Art Unit: 2193

### **Claim Limitations**

according to the first lexical context and without use of a structural command to the interactive software engineering tool.

### **Meaning of Claim Limitations**

The ability to automatically perform the transition with out the use of built in tools such as templates or pull down menus. The transition is performed by the interpreted functionality of the underlying software.

### **III. Art in Industry**

The mentioning of “JAVADOC” was made. In the event, JAVADOC is the product of the Assignee it would speed prosecution if the Applicant would submit an IDS and the original for sale and/or for use date and with a statement of relevance the differences between the claimed invention and the tool JAVADOC. The Examiner updated a search focusing on JAVADOC and has made of record a 1995 article by an employee of the Assignee (Sun Microsystems Inc.).

### **Applicant's Response to Interview Summary**

The statement that *“The Friendly article discloses a Java Compiler parsing comments into Java code to generate HTML markup documentation of the code comment. There is no disclosure or suggestion of an interactive software engineering tool that presents a user with behavior according to a corresponding set of lexical rules, or any of the subject matter of Applicant's claims.”*

***Claim Rejections - 35 USC § 103***

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

3. Claims 1- 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over JAVADOC as documented by “The Design of Distributed Hyperlinked Programming Documentation”, by Lisa Friendly, of Sun Microsystems Inc, 1995 in view of USPN #5,857,212 filed September 12, 1996 and issued on January 5, 1999 with an effective filing date of July 6, 1995. JAVADOC as described during the Interview of September 8, 2005, where the JAVADOC environment, supports the entry of two languages JAVA and HTML.

**Claim 1**

An interactive software engineering tool, which is embodied on a computer readable medium, that, for distinct portions of a single unit of source code thereof with behavior according to a corresponding set of lexical rules  
, wherein transition of the behavior from that in accordance with a first lexical context to that in accordance with a second lexical context is based on recognition of an opening boundary token according to the first lexical context and without use of a structural command to the interactive software engineering tool.

Examiner's Rejection

JAVADOC teaches software engineering tool (JAVADOC, page 1, JAVADOC as part of HotJava, Abstract), which is embodied on a computer readable medium (JAVADOC, page 1, Abstract, installed and executing compiler), that, for distinct portions of a single unit of source code ( JAVADOC, pages 14 – 15, Appendix A – Sample Source Code and Markup) thereof with behavior according to a corresponding set of lexical rules (JAVADOC, page 1, Abstract, parser requires lexical rules),  
, wherein transition of the behavior from that in accordance with a first lexical context (JAVADOC, page 14-15, Appendix A – JAVA), to that in accordance with a second lexical context ( JAVADOC, page 14 – 15, Appendix A, HTML ) is based on recognition of an opening boundary token according to the first lexical context and without use of a structural command to the software engineering tool (JAVADOC, page 15, Example – the comment marker is a boundary token “/\*\*” etc).

Art Unit: 2193

What JAVADOC does not teach is the “*interactive* software engineering tool”, where the user interacts with the tool in an “on the fly” environment. It is Van De Vanter who teaches the interactive software environment where “An editor for structurally represented computer programs transforms user-entered text on-the-fly into a stream of tokens that constitute words of the program under edit.” (Van De Vanter, Abstract, first sentence). Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to modify JAVADOC and add the interactive aspect, because interactive software engineering tools, “.. gives users great flexibility and minimizes learning costs.” (Van De Vanter, Col 1, lines 62 – 65).

**NOTE:** Van De Vander teaches the low level internals to the process claimed above.

### **Claim 2**

An interactive software engineering tool as recited in claim 1, wherein the behavior includes linguistically-driven typography.

#### Examiner's Rejection

As per the rejection for claim 1 and (Vander De Vander, provided by Pretty Print and features based on linguistic (or structural ) knowledge and , (Vander De Vanter, col 1, lines 25 – 42 and “Syntactic context” in part (Van De Vanter, col 7, lines 59- 64).

### **Claim 3**

An interactive software engineering tool as recited in claim 1, wherein the behavior includes lexical analysis of text based on a then operative one of the first and the second lexical contexts.

#### Examiner's Rejection

In view of claim 1 – (lexical analysis of text – as taught in claim 1) and the first lexical context (JAVA – in claim 1) and the second lexical context (HTML – in claim 1).

### **Claim 4**

An interactive software engineering tool as recited in claim 1, wherein the distinct portions are delimited by the opening boundary token and a corresponding, automatically-added closing boundary token.

#### Examiner's Rejection

In view of claim 1 and (Vander De Vanter, col 28, lines 63-66 – closing boundary token)

### **Claim 5**

An interactive software engineering tool as recited in claim 1, wherein the first and second lexical contexts respectively correspond to one of: a source language lexical context and a textual comment lexical context; a source language lexical context and a string literal lexical context; a source language lexical context and a character lexical context; and first and second source language lexical contexts.

#### Examiner's Rejection

See the example as taught in claim 1 – emphasis Appendix A.

Art Unit: 2193

### Claim 6

An interactive software engineering tool as recited in claim 1, wherein the single unit of source code is one of a line, statement or phrase; a function, procedure or method; and a markup language element, thereof.

#### Examiner's Rejection

As per the rejection for claim 1 and the **markup language element** (JAVADOC, page 4, HTML – Also the Abstract of JAVADOC, page 1).

### Claim 7

An interactive software engineering tool, which is embodied on a computer readable medium, that, in response to introduction of a language-defined opening boundary token at a cursor position in an edit buffer, automatically inserts a corresponding closing boundary token, such that display of edit buffer content past the cursor position maintains its pre-introduction association with a first lexical context and with linguistically-driven typography therefor, while subsequent entry at the cursor position is subject to a second lexical context.

#### Examiner's Rejection

JAVADOC teaches software engineering tool (JAVADOC, page 1, JAVADOC as part of HotJava, Abstract), which is embodied on a computer readable medium (JAVADOC, page 1, Abstract, installed and executing compiler), that, for distinct portions of a single unit of source code ( JAVADOC, pages 14 – 15, Appendix A – Sample Source Code and Markup) thereof with behavior according to a corresponding set of lexical rules (JAVADOC, page 1, Abstract, parser requires lexical rules), , wherein transition of the behavior from that in accordance with a first lexical context (JAVADOC, page 14-15, Appendix A – JAVA), to that in accordance with a second lexical context ( JAVADOC, page 14 – 15, Appendix A, HTML ) is based on recognition of an opening boundary token according to the first lexical context and without use of a structural command to the software engineering tool (JAVADOC, page 15, Example – the comment marker is a boundary token “/\*\*” etc).

What JAVADOC does not teach is the “*interactive* software engineering tool”, where the user interacts with the tool in an “on the fly” environment and detailed explicit teaching of the limitations, “in response to introduction of a language-defined opening boundary token at a cursor position in an edit buffer, automatically inserts a corresponding closing boundary token, such that display of edit buffer content past the cursor position maintains its pre-introduction association with a first lexical context and with linguistically-driven typography therefor, while subsequent entry at the cursor position is subject to a second lexical context.”

It is Van De Vanter who teaches the interactive software environment where “An editor for structurally represented computer programs transforms user-entered text on-the-fly into a stream of tokens that constitute words of the program under edit.”. (Van De Vanter, Abstract, first sentence) , in response to introduction of a language-defined opening boundary token at a cursor position in an edit buffer (Van De Vanter, col 27, lines 55-63, “a double quote is recognized as belonging to an incomplete string literal and Appendix A example as shown in claim 1) , automatically inserts a corresponding closing boundary token (Vander De Vanter, col 28, lines 63-66 – closing boundary token) , such that display of edit buffer content past the cursor position maintains its pre-introduction association with a first lexical context and with linguistically-driven typography therefor (Vander De Vander, provided by Pretty Print and



Art Unit: 2193

features based on linguistic (or structural ) knowledge and , (Vander De Vanter, col 1, lines 25 – 42 and “Syntactic context” in part (Van De Vanter, col 7, lines 59- 64 and (Van De Vanter, displayed by the typographical display processor; see column 28, lines 58-63); while subsequent entry at the cursor position is subject to a second lexical context (Van De Vanter, col 28, line 63 through col 29, line 2).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to modify JAVADOC and add the interactive aspect, because interactive software engineering tools, “.. gives users great flexibility and minimizes learning costs.” (Van De Vanter, Col 1, lines 62 – 65).

#### **Claim 8**

An interactive software engineering tool as recited in claim 7, wherein display of symbols entered into the second lexical context is in accordance with linguistically-driven typography distinct from that employed in the first lexical context.

##### Examiner's Rejection

As per the rejection for claim 1 and provided by Pretty Print and features based on linguistic (or structural ) knowledge (Vander De Vanter, col 1, lines 25 – 42 and “Syntactic context” in part (Van De Vanter, col 7, lines 59- 64).

#### **Claim 9**

An interactive software engineering tool as recited in claim 7, wherein lexical analysis of symbols entered into the second lexical context is in accordance with lexical rules distinct from that employed for the first lexical context.

##### Examiner's Rejection

As per the rejection for claim 7 and wherein lexical analysis of symbols entered into the second lexical context is in accordance with lexical rules distinct from that employed for the first lexical context (a double quote is recognized as belonging to an incomplete string literal; see column 27, lines 55-63 and Van De Vanter, col 28, lines 63-66 also the Friendly reference as see in Example in Claim 7, Appendix A, see close symbol for comments”\*/”).

#### **Claim 10**

An interactive software engineering tool as recited in claim 7, wherein the second lexical context is delimited by the opening and closing boundary tokens.

##### Examiner's Rejection

As per the rejection for claim 7 and claim 9.

#### **Claim 11**

An interactive software engineering tool as recited in claim 7, wherein the first and second lexical contexts respectively correspond to one of source language lexical context and a textual comment lexical context; a source language lexical context and a string literal lexical context; a source language lexical context and a character lexical context; and first and second source language lexical contexts.

##### Examiner's Rejection

Art Unit: 2193

The limitations are redundant to the limitations covered in claim 10. The example of Appendix A.

### **Claim 12**

A method of operating an interactive software engineering tool, the method comprising: rendering a display presentation corresponding to a unit of source code, said display presentation corresponding to at least a first lexical context operative at an insertion point; recognizing interactive entry of an opening boundary token at the insertion point; and in response to said recognition of said opening boundary token, creating a second lexical context operative for subsequent interactive entry at the insertion point, wherein the second lexical context is delimited by said opening boundary token ;and a position in the source code immediately following the insertion point, wherein said opening boundary token is a valid lexical token in accordance with one of the first and the second lexical context and not a nonlexical, structural command to the interactive software engineering tool.

#### Examiner's Rejection

JAVADOC teaches software engineering tool (JAVADOC, page 1, JAVADOC as part of HotJava, Abstract), which is embodied on a computer readable medium (JAVADOC, page 1, Abstract, installed and executing compiler), rendering a display presentation corresponding to a unit of source code ( JAVADOC, pages 14 – 15, Appendix A – Sample Source Code and Markup) display presentation corresponding to at least a first lexical context operative at an insertion point; recognizing interactive entry of an opening boundary token at the insertion point (JAVADOC, page 14-15, Appendix A – JAVA), creating a second lexical context operative for subsequent interactive entry at the insertion point ( JAVADOC, page 14 – 15, Appendix A, HTML) , wherein the second lexical context is delimited by said opening boundary token (JAVADOC, page 15, Example – the comment marker is a boundary token “/” etc). ;and a position in the source code immediately following the insertion point, wherein said opening boundary token is a valid lexical token in accordance with one of the first and the second lexical context ( JAVADOC, the JAVA Source Code as per the example in Appendix A above) and not a nonlexical, structural command to the software engineering tool (JAVADOC, the “Source code comments” as per the example in Appendix A above).

What JAVADOC does not teach is the “*interactive* software engineering tool”, where the user interacts with the tool in an “on the fly” environment. It is Van De Vanter who teaches the interactive software environment where “An editor for structurally represented computer programs transforms user-entered text on-the-fly into a stream of tokens that constitute words of the program under edit.”. (Van De Vanter, Abstract, first sentence). Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to modify JAVADOC and add the interactive aspect, because interactive software engineering tools, “.. gives users great flexibility and minimizes learning costs.” (Van De Vanter, Col 1, lines 62 – 65).

### **Claim 13**

A method as recited in claim 12, further comprising: in response to said recognition of said opening boundary token, automatically inserting at said position in the source code immediately following the insertion point, a closing boundary token.

#### Examiner's Rejection

Art Unit: 2193

In view of claim 12, opening boundary token (JAVADOC, page 15, Example – the comment marker is a boundary token “/” etc) and (Vander De Vanter, col 28, lines 63-66 – closing boundary token).

**Claim 14**

A method as recited in claim 12, wherein stylistic rules applied to rendering of symbols within the second lexical context differ from those applied to rendering of symbols within the first lexical context.

Examiner's Rejection

See the example as taught in claim 12 – emphasis Appendix A – the symbol for entry of a comment in source code.

**Claim 15**

A method as recited in claim 12, wherein lexical rules applied to recognition of tokens within the second lexical context differ from those applied to recognition of tokens within the first lexical context.

Examiner's Rejection

See the example as taught in claim 12 – emphasis Appendix A – the symbol for entry of a comment in source code.

**Claim 16**

A method as recited in claim 12, wherein the first lexical context is a programming language lexical context; wherein the second lexical context is string literal lexical context; and wherein the opening boundary token is a quote (") character.

Examiner's Rejection

In view of the rejection for claim 12 and Van De Vanter, col 28, lines 8 – 24).

**Claim 17**

A method as recited in claim 12, wherein the first lexical context is a programming language lexical context; wherein the second lexical context is character lexical context; and wherein the opening boundary token is a single quote (') character.

Examiner's Rejection

In view of the rejection for claim 12 and Van De Vanter, col 28, lines 55 – 60).

**Claim 18**

A method as recited in claim 12, wherein the first lexical context is a programming language lexical context; wherein the second lexical context is textual comment lexical context; and wherein the opening boundary token is one of: a multiple line comment token (/\*); a single line comment token (//); and a document type comment token (/\*\*).

Examiner's Rejection

The first and second lexical content of JAVA and the second HTML as per the rejection for claim 12 in view of the examples of JAVADOC tags page 9 in section 5.2.2.

Art Unit: 2193

**Claim 19**

A method as recited in claim 12, wherein the first and second lexical contexts correspond to respective programming language lexical contexts.

Examiner's Rejection

As per the rejection of claim 12, and the exemplified in Appendix A in claim 12, first being JAVA and second being the comments in the source code.

**Claim 20**

A method as recited in claim 12, wherein at least one of the first and second lexical contexts is a markup language lexical context.

Examiner's Rejection

As per the rejection for claim 12 and the **markup language element** (JAVADOC, page 4, HTML – Also the Abstract of JAVADOC, page 1).

**Claim 21**

A method as recited in claim 12, wherein transitions between the first and second lexical contexts are performed in response to navigation events and in response to entry of valid lexical tokens such that the transitions are transparent to a user of the interactive software engineering tool.

Examiner's Rejection

In view of the rejection for claim 12, col 1, lines 25 – 27 (cut and paste are navigational events). Transitions are transparent (an De Vanter, col 4, lines 46 – 52 – the “recognized” which performs alignments is transparent as are the structures of Figure 3).

**Claim 22**

A method as recited in claim 12, wherein transitions between the first and second lexical contexts are performed in response to navigation events and in response to entry of valid lexical tokens such that a user of the interactive software engineering tool need not employ structural commands therefor.

Examiner's Rejection

As per the rejection of claim 21 – detailed in col 4 section).

**Claim 23**

A method as recited in claim 12, wherein the interactive software engineering tool includes one or more of an editor; a source-level debugger; and a source analyzer.

Examiner's Rejection

As per the rejection for claim 12 and the editor (JAVADOC, page 12, “...editing source code...”).

**Claim 24**

A method as recited in claim 12, wherein said unit of source code includes one or more of: a line; a statement; a **markup language element**; and a function or procedure.

Examiner's Rejection

Art Unit: 2193

As per the rejection for claim 12 and the **markup language element** (JAVADOC, page 4, HTML).

### **Claim 25**

A computer program product. encoded in at least one computer readable medium and comprising: functionally-descriptive encodings of at least first and second language contexts; and instructions at least partially implementing a source, code editor that invokes the second language context nested within the first language context based solely on recognition of a boundary token defined by the first language context and entered at the cursor position, while maintaining pre-existing language context past the cursor position.

#### Examiner's Rejection

JAVADOC teaches software engineering tool (JAVADOC, page 1, JAVADOC as part of HotJava, Abstract functionally-descriptive encodings of at least first ( JAVADOC, pages 14 – 15, Appendix A – Sample Source Code and Markup) and second language contexts ( JAVADOC, page 14 – 15, Appendix A, HTML);

What JAVADOC does not teach is the underlying architecture functionally-descriptive encodings of at least first and second language contexts. It is Van De Vanter who teaches the the underlying architecture functionally-descriptive encodings of at least first and second language contexts (as provided by the following). and instructions at least partially implementing a source, code editor (Van De Vanter, Abstract, first sentence). that invokes the second language context nested within the first language context (JAVADOC, page 1, Abstract, installed and executing compiler), and entered at the cursor position (Van De Vanter, Abstract and Summary of Invention , col 4, lines 27-37 ), while maintaining pre-existing language context past the cursor position (Van De Vanter, Abstract and Summary of Invention , col 4, lines 27-37 and Figure 5, #158d).

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to modify JAVADOC and implement the underlying architecture functionally-descriptive encodings of at least first and second language contexts of Van De Vanter, because, editors with functionally-descriptive encodings of at least first and second language contexts, “.. gives users great flexibility and minimizes learning costs.” (Van De Vanter, Col 1, lines 62 – 65).

### **Claim 26**

The computer program product of claim 25, embodied as one or more of: **an editor**; a source-level debugger; and a source analyzer.

#### Examiner's Rejection

As per the rejection for claim 25 and the editor (JAVADOC, page 12, “ ...editing source code...”)

### **Claim 27**

The computer program product of claim 25, embodied, at least in part, as a language specialization component for integration with a software engineering tool.

#### Examiner's Rejection

Art Unit: 2193

As per the rejection for claim 25 and (JAVADOC, page 4, API – Application Interface Program in section 3. Design Requirements and JAVADOC, page 8, “5. Extensions to JAVADOC).

**Claim 28**

The computer program product of claim 25, supplied, at least in part, via a communications medium for execution on a computer coupled thereto.

Examiner's Rejection

As per the rejection for claim 25 and (JAVADOC, page 4, the inherent *communications medium* required for “Distributed On the World-Wide Web”).

**Claim 29**

The computer program product of claim 25, wherein the at least one computer readable medium includes at least one of magnetic storage medium, optical storage medium, electronic storage medium, a network medium, wireline medium, wireless medium or other communications medium.

Examiner's Rejection

As per the rejection for claim 25 and (JAVADOC, page 4, the inherent *network medium* required for “Distributed On the World-Wide Web”).

**Claim 30**

A computer system comprising: a display; memory; a language-based editor program executable thereby; and a buffer defined by the source code editor program and instantiable in the memory, wherein the language-based editor program renders contents of the buffer to the display in accordance with an associated language context, and wherein the language-based editor program recognizes entry of a transitional opening token defined by a first language context and, in response thereto, associates text subsequently entered into the buffer at an insertion point thereof with a second language context, while maintaining a pre-existing association between the first language context and contents of the buffer past the insertion point.

Examiner's Rejection

JAVADOC teaches software engineering tool (JAVADOC, page 1, JAVADOC as part of HotJava, Abstract), with two language contexts ( JAVADOC, pages 14 – 15, Appendix A – Sample Source Code and Markup) wherein the language-based editor program renders contents of the buffer ( **buffer** - called “token stream segment” – col 23 line 32) to the display in accordance with an associated language context (Van De Vanter, col 23, lines 30 to Col 24, lines 1 – 55- Inserting a Character and col 23, lines 30 to Col 24, lines 1 – 55- Inserting a Character and col 4, lines 39 - 59), and wherein the language-based editor program recognizes entry of a transitional opening token defined by a first language context (Van De Vanter, col 31, lines 12 to 35, - JAVA in the JAVADOC example Appendix A) and , in response thereto, associates text subsequently entered into the buffer at an insertion point thereof with a second language context (Van De Vanter, col 31, lines 12 - 35, - Comments in Source code in the JAVADOC example Appendix A), while maintaining a pre-existing association between the first language context and contents of the buffer past the insertion point (Van De Vanter, col 4, lines 39-59 , a position corresponding to the insertion point 157).

Art Unit: 2193

Therefore, it would have been obvious to one of ordinary skill in the art at the time of invention to combine the teachings of JAVADOC and Van De Vanter, “.. gives users great flexibility and minimizes learning costs.” (Van De Vanter, Col 1, lines 62 – 65).

### ***Response to Arguments***

4. The rejection of claims 1 – 30 has been withdrawn. Applicant’s arguments filed September 28, 2005 have been fully considered.

### **Applicant’s Remarks to Interview Summary**

The statement that *“The Friendly article discloses a Java Compiler parsing comments into Java code to generate HTML markup documentation of the code comment. There is no disclosure or suggestion of an interactive software engineering tool that presents a user with behavior according to a corresponding set of lexical rules, or any of the subject matter of Applicant’s claims.”*

### **Examiner’s Response to Remarks about Friendly Reference**

The Friendly reference documents Javadoc which discloses a Java compiler parsing comments in Java code to generate HTML markup documentation of the code comments. The Applicant states the limitation “an interactive software engineering tool that presents a user with behavior according to a corresponding set of lexical rules...”. The Friendly reference appears to be a two step process of writing and submit to a compiler. The invention is a one step interpreted process. When the Applicant states the Friendly reference does not teach “an interactive software engineering tool that presents a user with behavior according to a corresponding set of lexical rules...”, the key to this statement to be true is the interactive portion is missing from Javadoc. The abilities of Javadoc as mentioned in the interview are supported by the Friendly reference

Art Unit: 2193

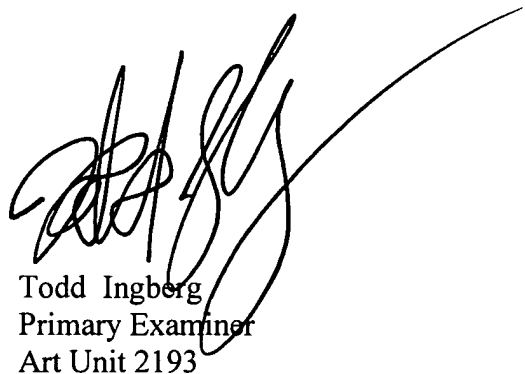
and Applicant's statements. The functionality appears the same with the exception of the "interactive" aspect.

### ***Correspondence Information***

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Todd Ingberg whose telephone number is (571) 272-3723. The examiner can normally be reached on during the work week..

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571) 272-3719. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).



Todd Ingberg  
Primary Examiner  
Art Unit 2193